ATLAS Grid Workload on NDGF resources: analysis, modeling, and workload generation

Dmytro Karpenko[1,2], Roman Vitenberg[2], and Alexander L. Read[1]

[1]Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
[2]Department of Informatics, University of Oslo, P.b. 1080 Blindern, N-0316 Oslo, Norway

***Abstract*—Evaluating new ideas for job scheduling or data transfer algorithms in large-scale grid systems is known to be notoriously challenging. Existing grid simulators expect to receive a realistic workload as an input. Such input is difficult to provide in absence of an in-depth study of representative grid workloads.**

In this work, we analyze the ATLAS workload processed on the resources of NDG Facility. ATLAS is one of the biggest grid technology users, with extreme demands for CPU power and bandwidth. The analysis is based on the data sample with ~1.6 million jobs, 1,723 TB of data transfer, and 873 years of processor time. Our additional contributions are (a) scalable workload models that can be used to generate a synthetic workload for a given number of jobs, (b) an open-source workload generator software integrated with existing grid simulators, and (c) suggestions for grid system designers based on the insights of data analysis.

I. Introduction

Since their introduction a couple of decades ago grid-based systems have been gradually growing in scale (number of cores and sites, storage requirement and capacity, amount of utilized bandwidth) as well as complexity of job scheduling and data distribution. One example of a large-scale grid system is the data processing and dissemination infrastructure of the ATLAS project [30] at the European Organization for Nuclear Research (CERN) whose goal is to test new theories of particle physics and observe phenomena that involve high energy particles using the Large Hadron Collider (LHC). The experimental data are processed, stored and disseminated to about 150 universities and laboratories in 34 countries. The infrastructure needs of the project in 2010 were estimated to be 112.2 MSI2k of computing power and 106 PB of storage. It is one of the largest infrastructures for cooperative computation in the world in terms of the geographic distribution, federation effort, number of users, etc.

In the advent of such systems, there is a significant body of research on improving algorithms and strategies, e.g., for job scheduling and data transfer [8], [10], [23]. However, the challenge of evaluation is a major impediment towards fast introduction of new ideas. The deployment scale does not allow testing proposed improvements in the actual production environment in a non-intrusive manner. While the production environments of systems such as ATLAS are typically accompanied by experimental clusters, these clusters cannot sustain the actual large-scale workload and their availability to the research community is limited.

SC12, November 10-16, 2012, Salt Lake City, Utah, USA
978-1-4673-0806-9/12/$31.00 © 2012 IEEE

It is therefore not surprising that most new ideas for scheduling and data distribution in grid-based systems are evaluated by simulation. There have been a number of popular simulators developed for grid systems [6], [9], [17], [21], [27]. At the same time, the question of workload that should be used in the simulation remains largely unresolved. Existing simulators expect the user to provide a workload as input. Some of the simulators are capable of producing synthetic workloads following simple statistical distributions, which are not necessarily representative of the workloads in actual grid systems. While some of the systems make workload traces available [12], these traces are incomplete, e.g., with respect to per-job data transfer information or data sharing across the jobs. Furthermore, it is challenging to scale these traces with respect to the size of the sample in order to generate a desired number of jobs without building a comprehensive workload model.

In this paper, we analyze a significant sample of the ATLAS workload that has been processed on the resources of Nordic Data Grid Facility (NDGF) [5] and build a data model for it. The collected data sample that lays the grounds for the analysis spans the period of 6 months and includes over 1.6 million jobs that transferred a total of 1,723 TB of data and used 873 years of processor time. To the best of our knowledge, the provided analysis is the first to consider per-job data transfer information at the granularity of individual files as well as correlation between various workload parameters including data sharing. As the analysis shows, many of the parameters cannot be accurately described by simple statistical distributions, which led us to develop a custom model based on variable interval histograms for estimating the density of each parameter.

We have released an open-source workload generator software based on the data model [1]. The generator is able to produce workload traces in the extended SW [16] or GWA [12] formats, which are standardized workload formats in this area. The generated samples can be used as input in simulations to test new data transfer, caching, scheduling, and job management policies in grid.

Finally, the analysis allows us to derive a number of insights and provide suggestions for grid system designers, e.g., with respect to data caching as well as data transfer policies. Since the communication between different sites is

loosely coupled and limited in terms of bandwidth, it makes the study of data transfer in ATLAS so interesting in our opinion. The experience of the grid production that the ATLAS collaboration possesses is of value for the design of any new large-scale and complex grid infrastructures.

II. Related Work

A. Workload analysis in grid computing

Collecting and analyzing traces of the workloads is regarded as an important research activity for a broad variety of complex distributed systems, with the purpose of reproducibility in simulations. In this section, we focus on past analysis performed in the area of parallel and grid computing.

Parallel Workloads Archive (PWA) [2] presents a comprehensive collection of workloads for parallel and grid systems. The maintainers have developed a standardized parallel workload (SW) format [16], in which most logs are stored. A record about each job includes 18 fields (some of which are optional) specifying meta-information about the job as well as a variety of resources the job requires. In addition to raw logs from a large number of parallel execution environments, the archive includes several workload models, both derived from the logs and following standard statistical distributions. While these models have subsequently been used in a large number of works on parallel computing, they do not capture specific information related to grid jobs and are not necessarily representative of grid workloads.

Grid Workloads Archive (GWA) [12] and Grid Observatory (GO) project [19] follow the practice established by PWA but focus exclusively on grid-related computing traces. GO offers a large collection of traces, including detailed status tracking of grid jobs and storage elements logs. However, the traces do not provide per-job transfer information (the number of input and output files, download location, size of transfer data, etc.), nor is it possible to match jobs traces with storage traces because the latter are anonymized.

In order to include more specific and detailed information about these traces, the creators of the GWA extended the SW format by adding 13 extra fields to the description of each job. However, even this extended format does not allow for specifying data transfer requirements of a job in terms of input and output files. An attempt to fit the trace data into standard non-custom distributions has been made in [11] and [20].

The information provided by PWA, GWA, and GO has been used in a significant body of research. For example, the traces from PWA are used as an input to simulation in [4], [28] and [31]. The workloads from GWA are used for testing the dependency searching methodology proposed in [15], and the data grom GO serves as the basis for developing job resubmission model in [25].

In this paper, we propose to extend the GWA format with information about job requirements for individual units of data. The workload generator described in Section VI outputs workloads in this extended format.

B. Grid simulation

One of the widely used general-purpose grid simulators is SimGrid [9], a framework for evaluating cluster, grid, and peer-to-peer algorithms and heuristics. SimGrid supports simulation of resources based on defined CPU and bandwidth capabilities. It does not consider disk space. The data transfer simulation methodology of SimGrid is presented and analyzed in [18].

Another widely used general-purpose grid simulator is GridSim [6]. GridSim is capable, like SimGrid, of simulating computational grids. In addition, a number of specific extensions presented in [14] enable it to model data grids, i.e. grids that manage large amounts of distributed data.

While SimGrid and GridSim are used for evaluating a large variety of grid applications and techniques, they rely on the user to specify the workload on which the application is tested. In particular, they are not compatible with the GWA format. In Section VI, we consider how to adopt the methodology for data transfer simulation in GridSim in order to utilize the output produced by our workload generator.

The GWA analysis work was also expanded with a simulation framework DGSim [21]. DGSim is capable of simulating a grid environment (or even several interconnected environments) using workflows that are based on real-world models. The simulator additionally possesses features such as adaptively varying the number of processor cores in the environment or simulating their failures. At the same time, DGSim does not include simulation of data transfer.

In addition to general-purpose grid simulators, there exists a number of specialized simulation tools in grid. For example, OptorSim [17] and ChicagoSim [27] were constructed for the purpose of studying the effectiveness of a predefined set of replication algorithms and protocols within data grid environments. Such tools naturally use a more restricted input format compared to general-purpose grid simulators, which diminishes their potential for interoperability with an SW- or GWA-based format, such as the one we propose in this paper.

C. Data transfer in grid

The importance of considering data movement in grid has been stated in a significant body of research, e.g., in [3], [24], and [27]. These works focus on enhancing data movement strategies. In [26], the authors propose a grid data transfer framework with scheduling capabilities and explore how different transfer strategies impact the performance in grid. The workload generator proposed in our paper endows all of the above works with the ability to evaluate the proposed methodologies on representative grid workloads.

A number of publications briefly consider data transfer as part of a comprehensive umbrella for simulated workflows. For example, in [7] each job is allowed to have a single stream of input data that is not split into files and that cannot be shared with the streams for other jobs. The framework does not include provision for output data. In [29] a transfer model of input data is proposed, with a set of tasks sharing a common set of files. Similarly to [7], the model does not consider output

data. Furthermore, the model does not allow for varying sizes of input files in a controlled fashion: The file sizes are assigned dynamically according to the purpose of each experiment, and never exceed 64MB. In our analysis in Section IV, we show that files of a larger size are typical in grid applications.

III. Data Collection Framework

The data which was the source for our work has been collected on the resources of NDGF [5]. NDGF is a collaborative effort between the Nordic countries (Norway, Sweden, Denmark and Finland) and is a resource provider for scientific communities in these countries. NDGF unifies 14 production sites, with over 5,000 cores assigned exclusively to grid computing, and many more available cores shared with non-grid jobs. NDGF has an accounting system that consists of a plugin on each NDGF site. The plugin registers information about each input and output file transfer (including the size and unique id of each file) performed by every grid job that successfully completes its execution. This allowed us to extract detailed per-job data transfer information. The extracted workload provides the basis for our work.

By far the largest user of NDGF resources is the ATLAS experiment [30]. ATLAS does research on physical phenomena at the LHC that were not observable at earlier, lower energy accelerators. The ATLAS collaboration comprises 150 universities and laboratories in 34 countries[1], all of which demand access to the processed and stored experimental data. To achieve this, the project in 2010 was estimated to need 112.2 MSI2k of computing power and 106 PB of storage. According to the ATLAS computing model [22], the ATLAS production happens at different levels, or tiers. Tier-0 at CERN is responsible for collecting all the raw data and distributing it to 11 Tier-1 centers around the world. The Tier-1 centers perform the primary processing of the data and distribute it further to numerous institutional-level Tier-2 centers. Tier-3 level, that gets input data from the Tier-1 and Tier-2 centers, is the level of end-user analysis; it may not even use grid and happen at user desktops. NDGF provides one of the 11 Tier-1 centers [13] and correspondingly receives 5% (about 40k jobs per day) of all ATLAS jobs. According to the ATLAS policy, each job is sent for processing to a site selected at random, with probabilities being assigned according to the amount of available resources[2]. Therefore, NDGF clusters are not assigned a particular type of jobs; they receive a proportional share of all the ATLAS jobs that circulate in the system. ATLAS production in NDGF is implemented in such a way that there is a clear separation between different job execution stages: input data transfer stage is followed by proper job execution, which is followed by output data transfer. It also has to be noted that the ATLAS workload currently includes very few parallel jobs, so that each job only requires one core.

[1]The figures were reported back in 2004, but underwent only a slight increase since then.

[2]In a few specific cases non-random submission policies, such as dispatching jobs to already cached data, can be used.

TABLE I: The main characteristics of the collected workflow

Name	Value
Period	01 February 2011 - 31 July 2011
# of sites	9
# of cores	>12,000
# of jobs	1,674,600
# of input files	2,837,400
# of output files	3,574,100
Amount of transfer	3,029 TB

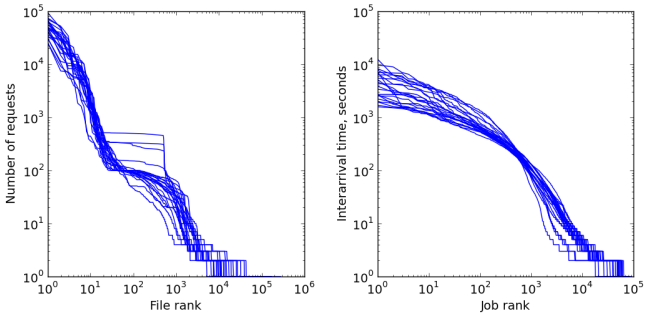
Table I summarizes the main characteristics of the collected workload. We consider only ATLAS jobs for the analysis, because they constitute the overwhelming majority of NDGF jobs and because majority of ATLAS jobs perform data transfer, that can be accounted by NDGF. We consider only the jobs that finished successfully, because failed jobs do not register complete information about themselves (e.g. walltime and upload requests), and only the jobs that performed data transfer, since our analysis is focused on data transfer in grid. Since Tier-3 level jobs are beyond the control of the grid infrastructure, our sample only comprises Tier-1 or Tier-2 level jobs.

The collected workload spans over the period of 6 months. It was processed by 9 NDGF Tier-1 and Tier-2 sites in Sweden, Norway and Denmark, employing more than 12,000 cores (the number of cores fluctuates slightly over time, but always stays above this value).

During this period the jobs requested 14,644,000 transfers of 6,411,500 files, reaching the total data transfer volume of 3 PB. Actual transfer, i.e. the requests that were not satisfied from cache, amounts to 7,299,700 transfers and to a volume of 1,723 TB. The jobs used 873 years of processor time.

Representativeness of the collected sample. The ATLAS production has been running since 2004. We gathered the information only for 6 months of its execution, so that it was important to make sure that the collected sample was representative of the entire ATLAS workload on NDGF resources, i.e. the parameters of the workload were stable over time.

To check the stability, we consider the fluctuation in various workload parameters across different weeks within the analyzed period. In Figure 1 the distributions for two of these parameters, namely file popularity and job interarrival time, are plotted on a per-week basis resulting in a curve for each week within the period. Plots for the other parameters are not shown due to the lack of space, but are available in [1]. As can be seen from the plot, the parameters are fairly stable and they exhibit a common pattern. The distributions for the other analyzed parameters do not change over time either. As an additional precaution, we extracted short samples from different periods outside our workload's span. Furthermore, we considered variations at a finer granularity, e.g., across different days in a week. We did not find any significant deviations: The parameters in these short samples follow the same distributions as in the total sample.



(a) File popularity (b) Job interarrival times

Fig. 1: Weekly distributions of parameters, logarithmic scale

IV. Analysis of the Workload

In this section we study the characteristics of the workload defined in Section III, with particular emphasis on data transfer. We separately consider the properties of files (with respect to their sizes and popularity) and jobs (with respect to transfer requests as well as interarrival and CPU times). In particular, we present the first, to the best of our knowledge, study of correlation between parameters of a grid workload from different categories, such as transfer and temporal characteristics. When considering the correlation between a pair of parameters, we produce a scatter plot and compute the Pearson coefficient. For the sake of readability, we opt to present distributions of individual parameters using survival functions and a logarithmic scale for the axes.

A. File characteristics

As mentioned in Section III, ATLAS jobs have distinct download and upload stages, that surround the proper execution. Furthermore, the output of the jobs in the analyzed sample is mostly destined for the Tier-3 level, and therefore is virtually never reused as input inside the sample. Therefore, we conduct separate analysis of input and output files.

Every input file can be requested by multiple jobs. Thus, an input file in the system can be characterized by its id, size and popularity (the number of requests for this file). Any file can potentially be replicated at different locations, but the replicas will have the same file id. Since a transfer request is issued for a particular file id rather than a replica, the physical location of a file is irrelevant for the analysis. During the analysis we do not distinguish if the input file was taken from cache or really downloaded, because we are interested in studying the distribution of file transfer requests among jobs and not the effectiveness of the caching algorithm.

In contrast, each output file is unique. Hence, it can be fully characterized by its file id and size.

In Figure 2 the survival functions of file size for input and output files are presented. File sizes are shown at the granularity of 1 MB, with values smaller than 1 MB converted to 1 MB. Observe that output files are significantly smaller on average compared to input files. This is due to the nature of ATLAS jobs, which tend to download a bigger amount of input raw data and produce a considerably smaller amount of

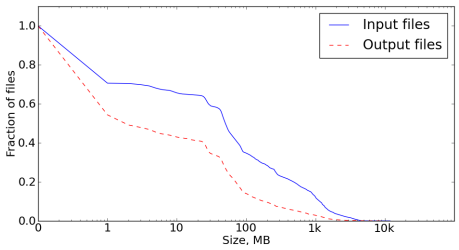
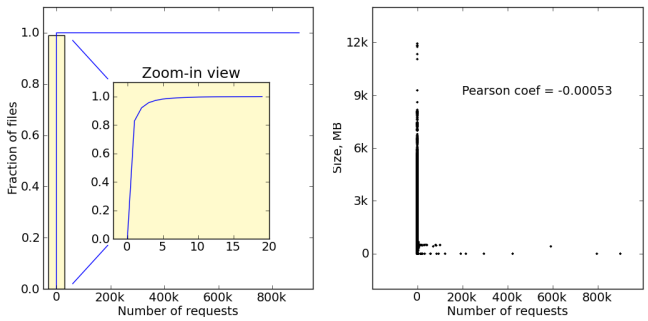


Fig. 2: Survival function of file size



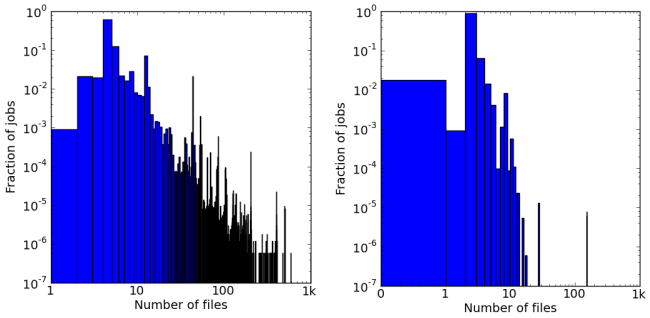
(a) CDF of file popularity (b) Popularity-size correlation

Fig. 3: Popularity of input files

processed data. However, the size of most input and output files is moderate, no more than 100 MB. Very small files of 1 MB or less constitute almost half of the output files and a quarter of the input files. The chance to observe a file bigger than 1 GB is 10% for downloads and just a few percents for uploads. When considering files larger than 1 MB, input files have a higher density in the range from several tens to several hundred MB, output files – from 30 to 100 MB. To sum up: the analyzed workload mostly includes files of small and medium size, input files tend to be larger than output files, files over 1 GB appear relatively rarely, but nevertheless appear.

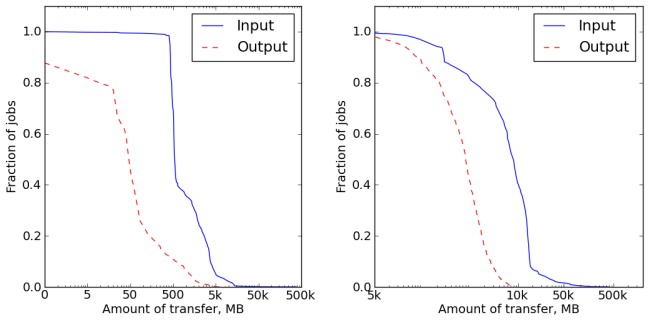
In Figure 3 the cumulative distribution of input file popularity and the correlation between the popularity and size of input files are plotted. Only 18% of input files are requested more than once, and slightly more than 1% of input files are requested more than 10 times. On the other hand, the files that are requested more than 10 times are responsible for 65% of transfer requests. Several files are requested more than 100,000 times. The most popular file is requested more than 900,000 times, or by 53% of the jobs. In a nutshell, the input files are divided into two groups: 98–99% of files are rarely, if ever, reused whereas 1–2% of files are reused so often that they constitute the majority of transfer requests.

The scatter subplot shows that there is practically no correlation between the file popularity and size, which is also confirmed by the close to 0 value of the Pearson coefficient. The figure clearly shows that the sizes of the files with low popularity, which form a large majority, are almost uniformly distributed. The correlation is slightly negative because popular files (with 60 or more requests) never have big sizes above 700 MB. Their average size is around 400–500 MB and files bigger than 500 MB are really rare.



(a) Input (b) Output

Fig. 4: Histograms of number of files per job, logarithmic scale



(a) Full range (b) Range 5GB–500GB

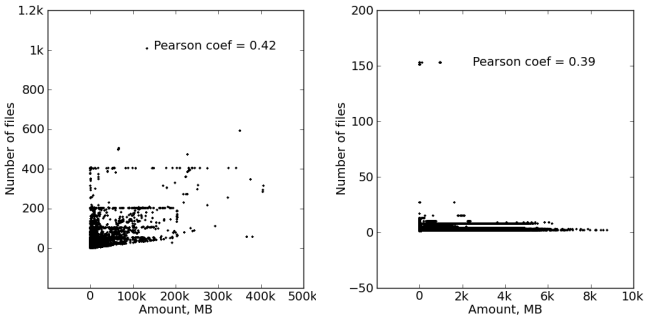
Fig. 5: Survival functions of total size of transfer per job

B. Job transfer characteristics

In Figure 4 the histograms of the number of input and output files per job are plotted. There are jobs that do not perform any upload and have 0 output files, so that the histogram for output files starts at 0 on the X-axis. There are as well some jobs that perform only upload without download, but they only constitute 0.05% of the total number of jobs in the workload compared to 1.75% of jobs that performed only download.

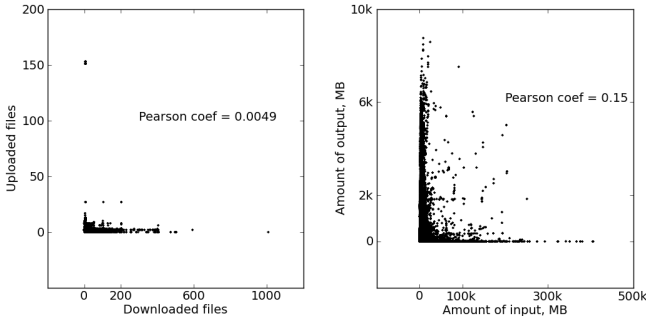
The most common number of download and upload requests per job were 4 and 2, respectively. There is also a fair number of jobs with 5, 12, and 43 input files. In the intervals between these numbers the density of jobs is distributed more or less evenly. Most jobs upload only a small number of output files whereas fewer than 100 jobs upload more than 13 files. While 2 output files per job is the most frequent case, there are many jobs that upload 3 files and a significant number of jobs that upload 0, 4, or 8 files.

In Figure 5 the survival functions of total size of input and output data per job are plotted. There are jobs that do not perform any upload, which explains why the survival function for output data starts at a value slightly below 1. The plot for input data shows that most jobs download a total amount of data that lies in the short range between 400 and 600 MB. In the upper interval, there is a fair number of jobs that download up to 10 GB in total. Jobs that download less than 400 MB or more than 10 GB are rare. The volume of output data per job is distributed much more smoothly. At the same time, the majority of the jobs have the total output size in the interval of 20–100 MB resulting in a clear abrupt descend in the plot.



(a) Input (b) Output

Fig. 6: Correlation between size of transfer and number of files per job



(a) Number of files (b) Size of transfer

Fig. 7: Correlation between download and upload per job

In Figure 6 the correlation between the transfer size and number of files per job is explored. The Pearson coefficient for both input and output data confirms the natural existence of a positive correlation. However, the correlation is not as strong as one might expect. The jobs that download most files are not those with the biggest input size, and the same observation holds for the uploads. Furthermore, some of the jobs that download/upload a relatively high number of files are characterized by a quite small transfer size. Yet, it can be clearly seen that the jobs with a small number of files never download a very large volume of input data.

In Figure 7 we examine correlation between input and output data characteristics. The value of the Pearson coefficient in Figure 7(a) indicates little correlation between the number of downloaded and uploaded files. While there is some positive correlation in terms of the total data volume downloaded and uploaded per job, this correlation is clearly weak. As with Figure 6, no job has the highest value for both parameters. For example, the jobs that download most files upload just a few.

It is challenging to provide a complete analysis of correlation in sharing input files by different jobs because of the number of combinations to consider (e.g., if a job requires files A and B while another job requires files B and C, does it increase the chances of both jobs sharing file D). Yet, we strive to capture the most significant dependencies in the workload model as described in Section V.

TABLE II: Parameters of walltime

Parameter	Overall	Clust.1	Clust.2	Clust.3	Clust.4	Clust.5
KSD	0	0.443	0.054	0.098	0.08	0.14
Mean	16,687	25,528	16,142	15,344	17,806	16,050
Median	16,214	21,403	15,918	14,910	16,205	9,611
STD	14,844	12,696	13,155	14,264	16,132	16,835
RMS	22,334	28,511	20,824	20,950	24,027	23,260

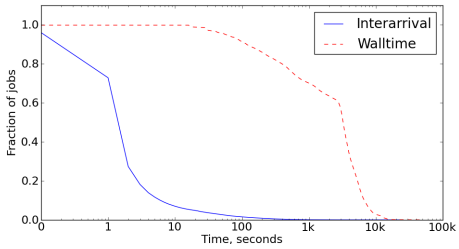


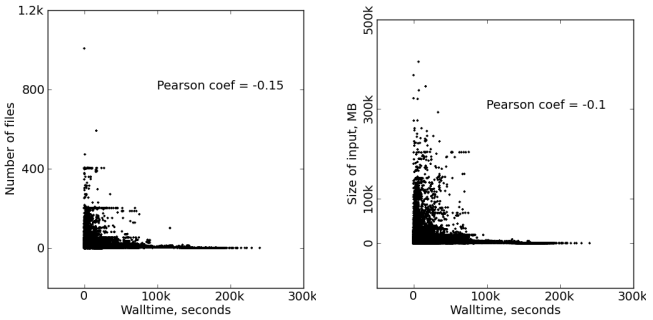
Fig. 8: Survival function of job walltime and interarrival time

C. Temporal characteristics of jobs

We consider two temporal job characteristics in our analysis, namely the distribution of interarrival and execution times. As mentioned in Section III, ATLAS production in NDGF is implemented in such a way that there is a clear separation between different job execution stages: input data transfer stage is followed by proper job execution, which is followed by output data transfer. This makes it easy to measure the proper execution time, which is equal to the walltime, i.e. the time the job is present on the execution host.

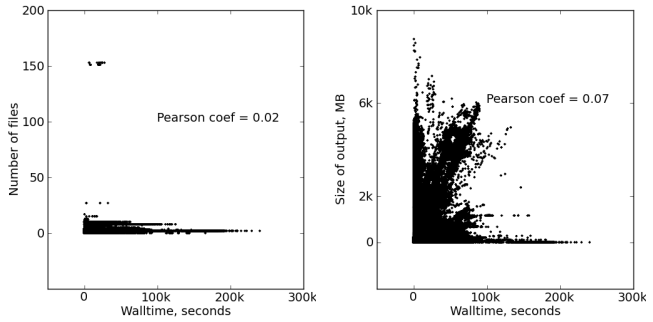
Of course, the absolute value of the walltime depends on the execution environment as much as on the actual jobs. In our analysis we are interested in the distribution of walltime and correlation with the other parameters rather than in the absolute value. Recall that NDGF consists of a number of sites. A potential issue with measuring the walltime is that the heterogeneity of the sites might skew the distributions. In order to verify this issue, we compare the parameters of the walltime distribution for the overall sample and for each of the 5 biggest clusters that accordingly receive most jobs. The parameters of interest include the Kolmogorov-Smirnov distance (KSD), mean, median, standard deviation (STD), and root mean square (RMS). The results presented in Table II indicate that the difference is generally insignificant. The bigger difference of values for Cluster 1 is caused not by a different architecture, but by the fact that this particular cluster slightly deviates from the random assignment policy and receives some percent of specialized jobs. This analysis allows us to conclude that the distribution of walltime for the overall sample is representative.

In Figure 8 the survival functions of job interarrival times and walltimes are plotted. There exist jobs that are submitted to the system virtually simultaneously, which explains why the survival function of interarrival times starts at a value slightly below 1. Most jobs arrive at short intervals, yet there have been observed pauses in production for up to several hours. The walltime plot shows that most of the jobs have running time in the interval from 40 minutes to almost 3 hours. Additionally,



(a) Number of files (b) Size of transfer

Fig. 9: Correlation between walltime and data input per job



(a) Number of files (b) Size of transfer

Fig. 10: Correlation between walltime and data output per job

there are many short jobs that finish in less than 15 minutes. On the other hand, the number of very long jobs with the walltime of several hours and more is almost negligible.

We also investigate if there is a correlation between the walltime of a job and its input data transfer characteristics. The results are shown in Figure 9. The correlation is present in both cases but at the same time in both cases it is quite weak. Interestingly, the correlation is negative so that the longer the job runs the less it downloads both in terms of the number of files and total size; this is also clearly visible on the plots. The explanation is due to the job semantics, namely the existence of a small number of pre-processing or administrative jobs that download a lot of data and then, either install new components in the system or rearrange the data by merging the files together or perform some other superficial pre-processing. At the same time, the jobs that perform thorough processing and analysis of physical data focus on a small or moderate number of relevant files.

We also consider possible correlation between the walltime and amount of output data. As shown in Figure 10, the observations are similar to those for the input data: Correlation is practically absent but jobs with the longest runtime never upload many files or a large volume of data.

D. Workload analysis summary

While some of the workload characteristics can easily be predicted, our analysis identifies a number of distinctive features. Firstly, weekly variations are insignificant and the parameters of the workload are stable over time. Secondly,

popular files are relatively few but they are responsible for 65% of transfer requests. Thirdly, correlation between most parameters is rather weak, though there are some interesting regularities: for example, the jobs with a long walltime or jobs with many input files only transfer a moderate amount of data.

V. Workload model

We now present a model for the workload described in Section IV. The main requirement for the model is to scale according to the target number of jobs. In other words, given the number of jobs, we should be able to generate the needed number of files shared across the jobs as well as derive the values for the other parameters so that all the distributions in the generated workload will be similar to those in the real data sample.

A. Methodology

In the beginning of the analysis we tried to build the model using only standard distributions. During the analysis it was, however, discovered, that almost none of the analyzed parameters can be approximated by standard probability distributions with an acceptable precision. Some of the parameters could be approximated by standard distributions, but only after applying a transformation (logarithmic or square root). Some other parameters could be reasonably well approximated only on a sub-range of the domain values. In principle, it might be possible to attain a good approximation by dividing the domain into a number of sections so that each section could be described by a different standard distribution. However, after having attempted this approach and realized that the division into subsections is not always obvious and that the resulting number of sections would be large, we opted to build custom distribution functions (CDF) instead.

The most precise way to model the workload would be to build a single multi-dimensional CDF describing the files, job input and output requests, job interarrival times and other workload parameters. Unfortunately, computing and even storing multi-dimensional CDFs in the main memory is prohibitively expensive because the amount of information grows exponentially with the number of dimensions. Our approach is to start by building separate distributions for files and jobs and subsequently create correlations and verify that they correspond to those in the actual sample.

The optimal number of bins in the histograms that describe various CDFs is also non-trivial to determine. Our first attempt to build histograms with a moderate number of bins resulted in real and synthetic data having the same shape and quite close distance, but sometimes very different mean and spread values. This indicates that the generated and real data had little overlap. We had to refine the granularity of all the CDFs and at the end we built a set of 111 CDF tables with a total of 1,197 entries. Due to the lack of space we cannot present them all in this paper; they can be found on the workload generator web page [1] along with additional information about our analysis. A sample of such a CDF is shown in Table III. As it can be seen from the table, the bins are often non-uniform. They were chosen after a close visual examination of the data by defining the CDF areas where the data had similar density. This allowed us to achieve a good balance with respect to the number of bins. The number of bins poses an additional challenge of visually presenting the histograms in a readable form. When plotting the histograms throughout this section, we usually limit the X-axis to the interval beyond which the density becomes close to zero.

TABLE III: Sample of a custom CDF function (number of input files per job)

Number of input files per job	CDF Value
3	0.042104
4	0.661795
5	0.785157
6–8	0.852227
9–11	0.874022

TABLE IV: Distribution parameters for # of files per job

(a) Input files

Name	Real	Generated
KSD	0.007	
Mean	6.613	6.6
Median	4.0	4.0
STD	9.431	9.073
RMS	11.519	11.219

(b) Output files

Name	Real	Generated
KSD	0.0005	
Mean	2.134	2.133
Median	2.0	2.0
STD	0.956	0.778
RMS	2.339	2.27

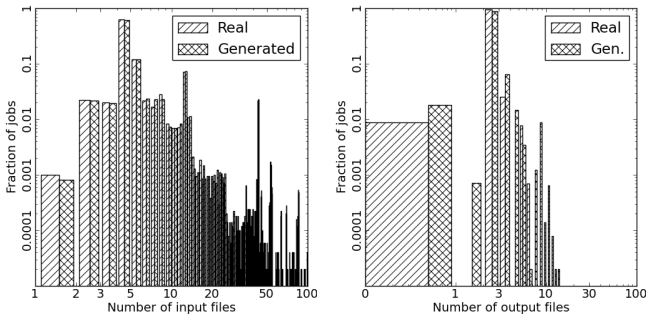
To determine how accurately we are able to reconstruct the data, we assess the goodness-of-fit by means of the Kolmogorov-Smirnov test, which computes the KSD between the generated (according to the model) and real samples. We also compare the main parameters of both samples: mean, median, STD, and RMS. While we tested our model and generated samples for various target numbers of jobs, we only present the results for 50,000 jobs in this paper. The real data sample is always used in its entirety.

B. Job transfer characteristics

The distribution for the number of input files per job can only be approximated by a custom CDF. The CDF was built using 48 non-uniform bins in the interval of 1 to 405 input files per job. There are only 35 jobs that downloaded more than 405 files. Beyond this threshold, the distribution for the real data sample becomes non-continuous, so that we set the upper limit for the model to 405. The parameters for the real and generated samples are shown in Table IV(a) and the comparative histogram is shown in Figure 11(a).

We choose the number of input files per job as the underlying characteristic for the model. We split the jobs into 48 categories (histogram bins) following the value of this parameter. All custom CDFs for the other job parameters are built separately for each of these 48 categories. This approach allows us to capture correlations between the number of input files per job and other job parameters at the granularity of individual categories.

The distribution of the number of output files per job cannot be easily approximated by any standard distribution either.



(a) Input files (b) Output files

Fig. 11: Histogram of number of files per job

TABLE V: Distribution parameters for input file popularity

Name	Real sample	Generated sample
KS Distance	0.02	
Mean	3.901	3.378
Median	1.0	1.0
STD	887.054	217.042
RMS	887.062	217.068

There are fewer than 100 jobs that uploaded more than 13 files each, so that we limit our distribution to at most 13 output files per job. We build a custom CDF on 14 equal intervals (from 0 to 13 upload files) for each category of jobs. Table IV(b) and Figure 11(b) present the results.

C. File characteristics

The distribution for the popularity of input files is similar to Zipf but it has a longer tail. Specifically, it can be reasonably well approximated by a Zipf distribution with the distribution parameter equal to 3, but only in the interval up to 30 requests. The interval of 31–60 requests can also be approximated by a Zipf distribution, but with some hand-tailoring, if we artificially increase the chance of appearance of numbers from 31 to 60. However, a Zipf distribution is not applicable starting from 61, since the sufficient quantity of frequently requested files cannot be drawn from it. Instead we split the input files into 6 categories: files with 100k and more requests; 10k–100k requests; 1k–10k requests; 100–1k requests; 60–100 requests; and fewer than 60 requests. The distribution of popularity in the last category is approximated by a slightly hand-tailored Zipf distribution with the parameter value of 3. For the 3-rd, 4-th and 5-th categories a custom CDF is built. For the first two categories it is impossible to build a reliable CDF because the number of files in both categories is low. Fortunately, scarcity of such files also means that the specific generation algorithm for these two categories does not affect the parameters of the overall distribution to any significant extent. This claim is corroborated by Table V and Figure 12 that compare the results for the actual and generated data, where the latter is produced using simply a random generation for the first two categories.

The difference between the parameters in Table V is explained by the fact that, when generating 50,000 jobs and the

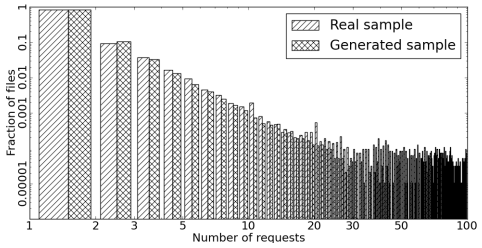


Fig. 12: Histogram of input file popularity

TABLE VI: Distribution parameters for file sizes

(a) Input files

Name	Real	Generated
KSD	0.033	
Mean	338.056	340.435
Median	51.0	52.0
STD	689.488	693.227
RMS	767.903	772.308

(b) Output files

Name	Real	Generated
KSD	0.045	
Mean	100.754	107.534
Median	2.0	2.0
STD	327.997	343.664
RMS	343.123	360.095

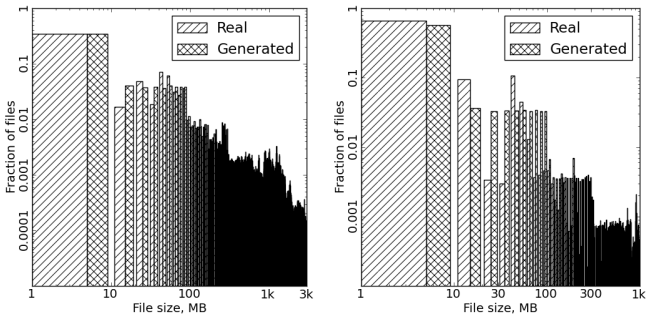
corresponding number of files, there will never be files that have the popularity of 100k or higher. Thus all the parameters of the generated sample become smaller, with the exception of median.

As mentioned in Section IV, there is some correlation between the size and popularity for highly popular files. Therefore, we create a separate custom CDF for size distribution for each of the 6 categories of files popularity. The results are presented in Table VI(a) and Figure 13(a).

Since the output files are unique, it is only needed to build a custom CDF for the distribution of output file sizes. The CDF is built for all output files at once, without breaking them into categories. The results are shown in Table VI(b) and Figure 13(b).

D. Size of data transfer per job

After we have generated the distributions for input and output files, the characteristics for data transfer per job have to come out as a result of assigning the generated files to jobs. Since the output files do not have a popularity characteristic, even a simple random assignment gives satisfactory results, as



(a) Input files (b) Output files

Fig. 13: Histogram of file sizes

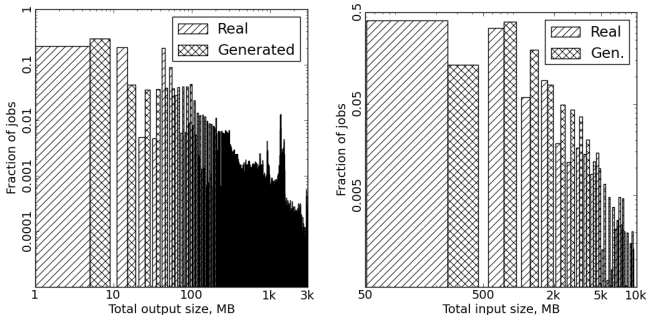
TABLE VII: Distribution parameters for total size per job

(a) Output

Name	Real	Generated
KSD	0.141	
Mean	214.526	223.152
Median	51.0	61.0
STD	524.393	488.672
RMS	566.577	537.212

(b) Input

Name	Real	Generated
KSD	0.329	
Mean	1,682.523	1,735.853
Median	541.0	926.0
STD	3,864.361	3,071.914
RMS	4,214.756	3,528.433



(a) Output (b) Input

Fig. 14: Histogram of total size per job

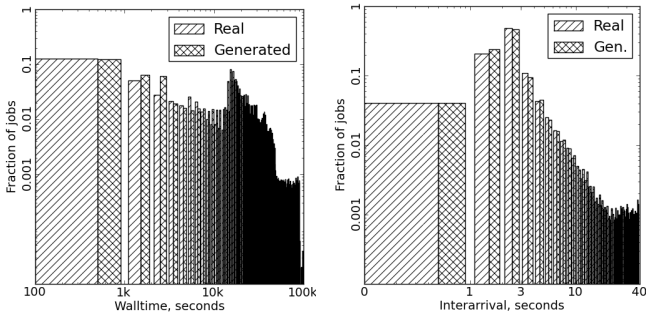
TABLE VIII: Distribution parameters for temporal characteristics of jobs

(a) Walltime

Name	Real	Generated
KSD	0.023	
Mean	16,687.237	17,175.119
Median	16,212.0	16,320.0
STD	14,847.473	15,866.482
RMS	22,336.323	23,382.257

(b) Interarrival time

Name	Real	Generated
KSD	0.003	
Mean	9.286	8.945
Median	2.0	2.0
STD	79.424	55.338
RMS	79.965	56.057



(a) Walltime (b) Interarrival time

Fig. 15: Histogram of temporal characteristics of jobs

illustrated by Table VII(a) and Figure 14(a). The distance is non-negligible and the parameters of the distributions slightly differ. Yet, the means are very close and the RMS values indicate a wide spread of the distributions. This implies that the real and generated samples, though not absolutely identical, exhibit a very significant overlap.

The input size per job is a more complicated characteristic because of a complex dependency on file popularity and size. To have a very close match between the synthetic and real data requires either copying of exact popularity and size values from the real sample or introducing an assignment algorithm with a strict control over the input size per job. The former approach is hardly acceptable when building a scalable model. The latter approach suffers from a prohibitively high computational complexity when generating a synthetic workload with tens of thousands of jobs and files. Our compromise solution is as follows. Earlier in this section, we define 48 categories of jobs and 6 categories of input files. For each category A of jobs and category B of input files, we compute the probability for a job from A to request a file from B. This allows us to attain a reasonable balance between control over the input size per job and workload generation speed. The results are presented in Table VII(b) and Figure 14(b). It can be seen that the distance between the samples is moderate yet bigger than for any other parameter we consider. However, the width of spreads and relative closeness of the mean values indicate that the overlap between the real and generated samples is acceptable. It must be noted that the parameters of the generated distribution exhibit some variance across different runs because the popularity and size of files change every time. While the typical KSD values range from 0.27 to 0.38, the distance can reach a value as high as 0.6 in very rare cases. Yet, even in such cases there is a good overlap between the samples, which makes our solution acceptable.

E. Temporal characteristics of jobs

Jobs walltime and interarrival times are generated according to custom CDFs that are built for the entire set of jobs without splitting them into categories. The results can be seen in Table VIII and Figures 15(a) and 15(b).

F. File sharing between the jobs

As we show in Section IV, there is relatively little correlation in most cases between the parameters we analyzed. However, there might be additional correlations that we did not specifically consider: if a job downloads a particular file, there might be a higher probability that it downloads an additional file. There may exist files that are only downloaded by a specific category of jobs, etc.

While the model does not explicitly capture this type of correlation, we verify that it does not produce a workload significantly different from the actual data sample in this respect. To this end, we use two indirect characteristics. Recall that we divide the jobs into categories (histogram bins) according to the number of input files. First, for each category A, we compute the ratio of the total number of unique input files downloaded by all the jobs in A to the number of jobs in A. Since every job in the same category downloads approximately the same number of files, it allows us to assess the amount of sharing between the jobs in this category. Second, for each category A, we compute the percent of input files shared with each other category B out of the total number of input files downloaded by all the jobs in A. This characteristic gives some insight into cross-sharing of input files between different categories.

While we do not present the computed values in this paper due to the lack of space, they can be found on the workload generator web page [1]. These values indicate that the difference between the model and real data sample is insignificant.

VI. Workload Generator

Based on the model presented in Section V we have released an open-source workload generator software publicly available for download from [1]. The generator is implemented as a Python script that accepts a target number of jobs as an input parameter and produces the given number of jobs and the corresponding number of input and output files. The job description is written in the extended GWA format, where the record for each job includes two additional fields: the list of indices for input and output files, respectively. The global lists of all input and output files are stored separately in a simple format: an index and a size are provided for each file. The generator makes use of the histograms, the mechanism for assigning files to jobs, and additional methodology described in Section V.

The generator is quite fast, using ca. 20 seconds to generate 50,000 jobs on a commodity desktop machine. At the same time the generator requires a lot of memory to complete this task (ca. 380 MB on our machines), which is mostly due to the large amount of histogram data as described in Section V. It is also a natural drawback of using the Python interpreter to run the script. On the other hand, since Python is a cross-platform language and we use only three external modules – "sys", "numpy", and "getopt" – that are available in any Python distribution, our generator can be used on almost any platform.

We performed some work to facilitate the use of the generator in conjunction with the GridSim simulator. In particular, we augmented the implementation of the configurator module in GridSim with the ability to read job traces that include lists of input and output files. The details and the implementation are available on the web site [1].

Future work on the generator includes several directions. For example, the generator can also be integrated with the SimGrid simulator; the code itself can be improved to run faster and use less memory; the generator can be integrated with other existing tools for mining and analyzing grid-related data.

VII. Lessons learned

The performed analysis allows us to derive a number of guidelines and recommendations for the benefit of grid system designers.

Replication and caching strategy: As shown in Section IV, 99% of the input files are rarely reused, yet the majority of the download requests is for the remaining 1%. This means that the benefits of caching popular files are tremendous whereas there is no point in caching most files. Since all popular files are of small or moderate size, a site cache does not have to be particularly big. At the same time, it makes sense to proactively identify and replicate popular files across all production sites. The analysis also has some bearing on the cache replacement policy. In particular, the Least Frequently Used policy would be problematic because if a once popular file is never later used because of an upgrade or a new physical experiment, it may practically take years before it is evicted from the cache.

Transfer prediction: The analysis shows the absence of variations in workload parameters over time, as seen in Figure 1 and explained in Section III. This calls for proactive prediction strategies while diminishing the motivation for reactive adaptation to an extent. Unfortunately, it is impossible to predict popularity of a file based on its static size. It is further difficult to predict the size of job transfer data based on the number of requested files or requested walltime. On the other hand, 60% of the jobs downloaded 4 files and uploaded 2, whereas 52% of all jobs have total input size between 400 and 600 MB and output size between 20 and 100 MB (45% of all jobs belong to both categories at the same time). This knowledge facilitates configuring data transfer implementation (i.e., the number of transfer threads per job and bandwidth allocation per job) in an effective way, at least for the majority of jobs. The threading policy, bandwidth and other parameters can also be adjusted for certain jobs outside this majority: the jobs with long requested walltime are not likely to transfer a lot of data; the jobs that have many transfer requests might transfer bigger volumes of data.

Since some of the input files are very large, it is important to break files into chunks rather than, e.g. allocating threads on a per-file basis. Furthermore, since the correlation between the number of data files per job and the total size of data is not so strong, the number of transfer threads per jobs should be allocated based on the latter and not on the former.

The average download to upload ratio for a job is 11 while the median ratio and the ratio for most of the jobs is close to 6. The knowledge of this range can govern the distribution of transfer threads as well as bandwidth allocation between uploads and downloads.

VIII. Conclusions

In this article we have reported on one of the biggest and most demanding grid workloads. We analyzed it, pointing out interesting facts and features, created a model for the workload, verified its correctness, and produced a tool that can generate synthetic workloads according to the model.

Future work encompasses several possible directions. There is a room for improving the model and making it more accurate, e.g., with respect to correlations and data sharing. The model can be extended by including additional parameters such as the job footprint or memory requirements, which are currently not recorded in the accounting database of NDGF. As described in Section VI, the generator can be improved in a variety of ways. Perhaps the most important topic for further research would be to improve state-of-the-art data transfer and job scheduling schemes for grid computing based on the provided analysis.

REFERENCES

[1] ATLAS grid workload on NDGF resources: analysis, modeling, and workload generation – auxiliary web-page. http://folk.uio.no/dmytrok/phd/ndgf_atlas_workload/index.html.

[2] Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[3] M. S. Allen and R. Wolski. The Livny and Plank-Beck problems: Studies in data movement on the computational grid. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, SC '03, pages 43–61, New York, NY, USA, 2003. ACM.

[4] V. Berten and B. Gaujal. Brokering strategies in computational grids using stochastic prediction models. *Parallel Comput.*, 33:238–249, May 2007.

[5] R. Buch, L. Fischer, O. Smirnova, and M. Groenager. The nordic data grid facility. *META (UNINEtt Sigma AS)*, 1:14–17, 2006.

[6] R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13):1175–1220, 2002.

[7] Y. Cardinale and H. Casanova. An evaluation of job scheduling strategies for divisible loads on grid platforms. In *Proceedings of the High Performance Computing & Simulation Conference*. HPC&S'06, 2006.

[8] A. Carpen-Amarie, M. I. Andreica, and V. Cristea. An algorithm for file transfer scheduling in grid environments. *CoRR*, abs/0901.0291, 2009.

[9] H. Casanova, A. Legrand, and M. Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, pages 126–131, Washington, DC, USA, 2008. IEEE Computer Society.

[10] E. Elmroth and P. Gardfjall. Design and evaluation of a decentralized system for grid-wide fairshare scheduling. In *Proceedings of the First International Conference on e-Science and Grid Computing*, E-SCIENCE '05, pages 221–229, Washington, DC, USA, 2005. IEEE Computer Society.

[11] A. Iosup et al. How are real grids used? The analysis of four grid traces and its implications. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, GRID '06, pages 262–269, Washington, DC, USA, 2006. IEEE Computer Society.

[12] A. Iosup et al. The grid workloads archive. *Future Gener. Comput. Syst.*, 24:672–686, July 2008.

[13] A. Read et. al. Complete distributed computing environment for a HEP experiment: experience with ARC-connected infrastructure for ATLAS. *Journal of Physics: Conference Series*, 119, 2008.

[14] A. Sulistio et al. A toolkit for modelling and simulating data grids: an extension to GridSim. *Concurr. Comput. : Pract. Exper.*, 20:1591–1609, September 2008.

[15] M. Lassnig et al. A similarity measure for time, frequency, and dependencies in large-scale workloads. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 43:1–43:11, New York, NY, USA, 2011. ACM.

[16] Steve J. Chapin et al. Benchmarks and standards for the evaluation of parallel job schedulers. In *Proceedings of the Job Scheduling Strategies for Parallel Processing*, IPPS/SPDP '99/JSSPP '99, pages 67–90, London, UK, 1999. Springer-Verlag.

[17] W. H. Bell et al. Simulation of dynamic grid replication strategies in OptorSim. In *Proceedings of the Third International Workshop on Grid Computing*, GRID '02, pages 46–57, London, UK, 2002. Springer-Verlag.

[18] K. Fujiwara and H. Casanova. Speed and accuracy of network simulation in the SimGrid framework. In *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, ValueTools '07, pages 12:1–12:10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] Grid Observatory. Web site. URL http://www.grid-observatory.org/.

[20] A. Iosup and D. Epema. Grid computing workloads. *IEEE Internet Computing*, 15:19–26, March 2011.

[21] A. Iosup, O. Sonmez, and D. Epema. DGSim: Comparing grid resource management architectures through trace-based simulation. In *Proceedings of the 14th international Euro-Par conference on Parallel Processing*, Euro-Par '08, pages 13–25, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] R. Jones and D. Barberis. The ATLAS computing model. *Journal of Physics: Conference Series*, 119, 2008.

[23] T. Kosar and M. Balman. A new paradigm: Data-aware scheduling in grid computing. *Future Gener. Comput. Syst.*, 25(4):406–413, April 2009.

[24] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, ICDCS '04, pages 342–349, Washington, DC, USA, 2004. IEEE Computer Society.

[25] D. Lingrand, J. Montagnat, J. Martyniak, and D. Colling. Optimization of jobs submission on the egee production grid: Modeling faults using workload. *J. Grid Comput.*, 8(2):305–321, 2010.

[26] W. Liu, B. Tieman, R. Kettimuthu, and I. Foster. A data transfer framework for large-scale science experiments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 717–724, New York, NY, USA, 2010. ACM.

[27] K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, HPDC '02, pages 352–359, Washington, DC, USA, 2002. IEEE Computer Society.

[28] R. Ranjan, R. Buyya, and A. Harwood. A case for cooperative and incentive-based coupling of distributed clusters. In *Proceedings of the 7th IEEE International Conference on Cluster Computing*. IEEE Computer Society Press, 2005.

[29] H. Senger, F. A. B. Silva, and W. M. Nascimento. Hierarchical scheduling of independent tasks with shared files. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '06, pages 51–59, Washington, DC, USA, 2006. IEEE Computer Society.

[30] The ATLAS Collaboration. ATLAS - A Toroidal LHC ApparatuS. Web site. URL http://atlas.web.cern.ch.

[31] C. S. Yeo and R. Buyya. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In *Proceedings of the 7th IEEE International Conference on Cluster Computing*, pages 27–30, 2005.